Séance 2 – Projet PENDU

1. Définition

Le jeu du PENDU consiste à proposer des lettres de l'alphabet pour deviner un mot. Le joueur dispose d'un certain nombre de vies qu'il perd chaque fois qu'il propose une lettre qui ne figure pas dans le mot à trouver(choisi aléatoirement dans une liste de mots).

Nous allons réaliser un jeu du PENDU en Python. Nous ajouterons des étapes progressivement.

C'est évident mais précisons-le : la réalisation doit être faite par vous, pas par un tuto ou une IA !

2. La source de données

Pour que le jeu soit motivant, nous avons besoin d'une liste de mots à deviner, celle-ci vous sera fournie pour gagner du temps. Le fichier qui contient cette liste est *liste_mots.py*.

Pour pouvoir accéder à cette liste, répondez à la question suivante :

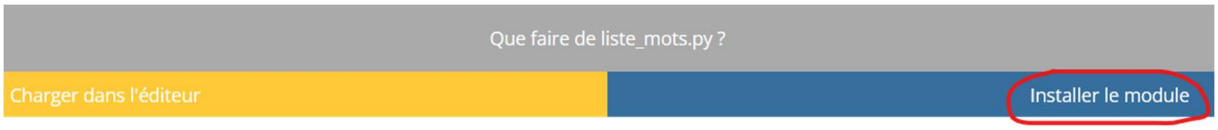
Qui dit liste de mots, dit type structuré, mais lequel est le plus approprié dans notre contexte ?
Liste, tuple, dictionnaire, ensemble ?

La bonne réponse est l'ensemble : il est préférable d'utiliser une liste sans doublons.
Notre ensemble s'appelle **mots**.
A noter qu'on utilisera les mots en majuscules sans accent, la liste fournie n'en contient pas.
Dans notre algorithme, nous exclurons, par prévention, les mots contenant des espaces ou des tirets.

Maintenant que vous avez pu télécharger le fichier *liste_mots.py*, il faut l'importer dans la console.

Pour l'utiliser dans la console Basthon, on procède de cette manière *:*

Rechercher le fichier sur votre ordinateur et choisir "Installer le module"



Le fichier est désormais accessible dans la console, nous pouvons accéder à l'ensemble mots grâce à cette commande :

```
from liste_mots import mots
```

3. Utilisation de bibliothèques Python

Python met à notre disposition un certain nombre de bibliothèques. Elles contiennent des fonctions utiles que nous n'aurons pas besoin de réécrire.

Recherchez dans la documentation https://docs.python.org/fr/3/library/index.html celles dont nous allons avoir besoin pour faire :

- une sélection aléatoire (d'un mot dans l'ensemble mots)

- des opérations communes sur des chaînes de caractères

Une fois que vous avez trouvé, il suffit de les importer :

import *nombibliothèque*

Nous avons maintenant configuré notre environnement de travail.

4. Choix du mot à faire deviner

Nous allons commencer par gérer la sélection du mot à trouver, en tirant un mot au hasard dans la liste.

Si vous avez bien importé random, il y a une méthode qui permet de renvoyer un élément aléatoire d'une séquence.

On ne souhaite pas récupérer un mot contenant un tiret ('-') ou un espace (' ').

Comment allez-vous procéder ?

5. Le pendu pas à pas

Réfléchissez aux étapes nécessaires pour réaliser ce jeu. Vous pouvez commencer par noter vos idées telles qu'elles viennent, ensuite il faudra les ordonner de manière logique et enfin vérifier leur enchaînement pour être certain que l'exécution soit la plus optimisée possible.

Il n'y a pas qu'une manière de réaliser cet algorithme, ne vous attendez pas à avoir tous la même progression !

Les questions que vous devrez vous poser :

- de quoi ai-je besoin ?
- jusqu'à quand mon algorithme doit-il fonctionner ?
- que dois-je traiter en priorité ?
- quelles sont les interactions avec le joueur ?
- quand doivent-elles avoir lieu ?

6. De quoi ai-je besoin ?

Faites la liste des variables, constantes, listes, tuples, etc. dont vous allez avoir besoin. Précisez systématiquement leur rôle (sous forme de tableau par exemple). Complétez cette liste au fur et à mesure de votre analyse.

- le mot à trouver et son masque ('- - - - - - - ')
- les lettres qui constituent le mot (sans doublon)
- les lettres de l'alphabet en majuscules (c'est là qu'on va avoir besoin de la bibliothèque string) pour comparaison et gérer les erreurs de saisie de l'utilisateur (bien-entendu sans doublon)
- la liste des lettres déjà proposées (sans doublon)

7. Jusqu'à quand mon algorithme doit-il fonctionner ?

Mon algorithme doit fonctionner jusqu'à ce que le mot soit trouvé et que le joueur ait perdu toutes ses vies.

Ou encore : tant que le mot n'est pas trouvé et qu'il y a encore des vies !

Et oui, la manière dont on pose le prédicat a son importance… A vous de choisir !

8. Que dois-je traiter en priorité ?

La gestion des vies (tentatives) peut être traitée en dernier.

On doit gérer (volontairement dans le désordre !) :

- récupérer la saisie proposée par le joueur (en majuscules)
- vérifier si elle fait partie du mot à trouver
- vérifier qu'elle n'a pas déjà été proposée
- vérifier qu'elle correspond à une lettre de l'alphabet
- l'affichage du mot sous la forme - - - - - avec les lettres placées au bon endroit au fur et à mesure qu'elles sont trouvées
- vérifier s'il reste des vies
- actualiser la liste des lettres proposées
- notifier le joueur (interactions à définir au point suivant).

9. Quelles sont les interactions avec le joueur ?

Le jeu n'ayant pas d'interface graphique, les interactions avec le joueur sont d'autant plus importantes. N'oubliez pas qu'il n'a qu'un écran noir et un curseur devant les yeux…

J'envisage 9 messages minimum, saurez-vous les retrouver ?

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____

10. Quand doivent-elles avoir lieu ?

Pour chaque message que vous avez numéroté ci-dessus, indiquez à quel moment il doit s'afficher :

1. _____
2. _____
3. _____

4. _____

5. _____

6. _____

7. _____

8. _____

9. _____

11. Composez le masque du mot à trouver

Nous allons devoir composer une espèce de masque de saisie qui va se compléter au fur et à mesure des propositions du joueur (si celles-ci sont bonnes !), comme on le ferait sur papier.

Rappelez-vous qu'une chaîne de caractères peut se parcourir comme une liste, un tuple, un dictionnaire, un ensemble.

On pourrait donc envisager une fonction qui parcourrai la chaîne de caractères lettre par lettre.

- Si la lettre a été trouvée, on l'affiche
- Sinon on la remplace par un tiret '-'

Cette fonction doit bien-sûr regarder si la lettre est présente dans le mot à trouver !

Pour les plus rapides d'entre vous : il est possible, en Python, de composer le masque en une seule instruction…